A MEASUREMENT STUDY OF 802.11N LINKS ON COMMODITY HARDWARE

BY

RAKESH KUMAR

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Professor Nitin H. Vaidya

ABSTRACT

802.11n is one of the recently ratified drafts for wireless networks. We modify an open-source wireless card device driver and conduct a performance analysis study. We find by measurements that the currently implemented algorithm in the open source device driver fails to successfully converge to the best physical layer rate under the given channel conditions. Several metrics to study the patterns of error are devised and measurements are taken in an office environment, at different times of day, and in an anechoic chamber. We find that 802.11n, which uses MIMO, presents a peculiar nature of performance in indoor and anechoic chamber environments and presents some interesting challenges as well as opportunities for better rate control algorithm design.

ACKNOWLEDGMENTS

Thanks to my parents, for their love and support; to my friends, who have been there through thick and thin; to my adviser, for his profound patience; to United States Department of State for supporting this study with a Fulbright fellowship; to the staffs of the Department of Electrical and Computer Engineering and Coordinated Science Lab at the University of Illinois, for providing me a nurturing environment; and finally, to all the teachers who imparted some of their wisdom.

TABLE OF CONTENTS

LIST OF TABLES v

LIST OF FIGURES vi

LIST OF ABBREVIATIONS vii

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 PRELIMINARIES 3
2.1 802.11n MAC Enhancements 3
2.2 Hardware and Ath9k Driver 6
2.3 Supported Physical Layer Rates 7
2.4 Anechoic Chamber 9

CHAPTER 3 RELATED WORK 12
3.1 Ath9k Rate Control Algorithm 12
3.2 Measurement Studies 13

CHAPTER 4 EXPERIMENTS 15
4.1 Enabling Process 15
4.2 Script Description 18
4.3 BA Bitmap Parameters 19
4.4 Measurement Results 20

CHAPTER 5 CONCLUSIONS AND FUTURE WORK 30
5.1 Summary of Measurements 30
5.2 Future Work 30

REFERENCES 32

LIST OF TABLES

2.1 HT20 Rate Table Employed During the Experiments 8
2.2 HT40 Rate Table Employed During the Experiments 9

3.1 Rate Filling Strategy Used by Minstrel During MRR 13

LIST OF FIGURES

2.1 Block acknowledgment mechanism in 802.11n 5
2.2 Ath9k architecture 7
2.3 Experiment conducted in the anechoic chamber 11

4.1 Driver architecture diagram for fixing physical layer rate . . . 17
4.2 Driver architecture diagram for exporting block acknowledgment bitmap statistics 18
4.3 Distribution of observed UDP throughput (Mbps) on the desktop link in office environment 21
4.4 Distribution of observed UDP throughput (Mbps) on the Netbook link in office environment 22
4.5 Distribution of observed UDP throughput (Mbps) on the netbook link in anechoic chamber 22
4.6 Distribution of fraction of bad frames in BA over desktop link in office environment 23
4.7 Comparison of average, normalized number of transitions in office and anechoic chamber 24
4.8 Comparison of average, normalized one burst size in office and anechoic chamber 24
4.9 Comparison of Distribution of highest throughput (Mbps) rates during day and night 25
4.10 Comparison of distribution of normalized, average number of transitions during day and night 26
4.11 Comparison of distribution of normalized, average one burst size during day and night 26
4.12 Comparison of distribution of normalized, average zero burst size during day and night 27
4.13 Comparison of distribution of observed UDP throughput (Mbps) between a single stream and dual stream rate 28
4.14 Comparison of distribution of normalized, average number of transitions between a single stream and dual stream rate . . 28
4.15 Comparison of distribution of normalized, average one burst size between a single stream and dual stream rate 29

LIST OF ABBREVIATIONS

A-MPDU	Aggregated MAC Packet Data Unit
ADDBA	Add Block Acknowledgment
BA	Block Acknowledgment
DELBA	Delete Block Acknowledgment
MAC	Medium Access Control
MRR	Mutli-Rate Retry
PPDU	Physical Layer Packet Data Unit

CHAPTER 1

INTRODUCTION

The widespread popularity of Wi-Fi (802.11) has motivated the development of higher throughput wireless networks. 802.11n is the set of amendments ratified for the 802.11 standard, for serving this need. The finalized draft mandates use of a range of enhancements such as MIMO at the physical layer and frame aggregation with block acknowledgments at the MAC layer. The standard draft does not, however, mandate usage of any particular rate control algorithm. There is a variety of rate-control techniques previously proposed for 802.11a/b/g networks, and Wong et al. [1] provide a comprehensive survey of their strengths and pitfalls.

Frame aggregation essentially means transmitting back-to-back, more than one frame at a time, whenever a transmit opportunity is presented. The experimental study conducted by Pelichrinis et al. [2] and the analytic results [3] indicate the essential nature of frame aggregation and block acknowledgment enhancements for reaching the performance targets 802.11n was designed to achieve.

However, the rate control problem in the specific context of 802.11n needs rethinking. There are broadly three classes of protocols available for 802.11, classified based on the type of information they harness to make rate control decisions for the succeeding transmission(s). There are those that look at the frame loss pattern for individual frame transmissions, i.e. in the absence of any aggregation [1], [4], [5]. These techniques use information about the success/failure of previous transmissions and use that to adapt rate. Then, there are techniques that use SINR as an information input. However, in the case of a MIMO PHY, the definition of SINR indicator becomes ambiguous and is not made available in public domain by the wireless card vendors. Finally, there are techniques that use soft decisions and measure of confidence in the received bits [6]. But these techniques are still only applicable to experimental systems, and the implemented versions are too slow to be used

for data rates as high as 802.11n has to offer.

This study explores the space of possible solutions to the rate control problem posed by currently deployable 802.11n networks. We use an open-source wireless card driver called ath9k, for performance analysis of 802.11n in general and of the rate control algorithm currently used by the driver, in particular. The current rate control algorithm does not take into account the pattern of error observed in the last transmission(s) to effectively modify the rate of the next transmission. The thesis pursued herein is that the pattern of the errors observed in the previous transmission is crucial because it represents the pattern of interference and noise observed in the environment.

Understanding pattern of interference is a hard analytic problem. Most of the analytical studies present for 802.11 a/b/g assume the interference process to be i.i.d. across nodes, and this assumption gives remarkably accurate results for large enough network sizes [7]. However, practical and experimental wireless network deployments have exhibited patterns in being exposed to interference [8]. Hence, this study measures some metrics to characterize the nature of errors observed and provides reasonable heuristics for the design of a rate control algorithm.

The rest of this thesis is organized as follows: Chapter 2 discusses the preliminaries and provides the necessary background for the study. Chapter 3 discusses related work. Chapter 4 discusses experimental setup, the process that enabled it, and results from the measurement studies we conducted. Chapter 6 speculates on future work and concludes the thesis.

CHAPTER 2

PRELIMINARIES

2.1 802.11n MAC Enhancements

Frame aggregation and block acknowledgment mechanisms are both enhancements introduced by 802.11n. The MAC protocol enhancements include two levels of aggregations, i.e. at the ingress (A-MSDU) and at the egress of MAC layer (A-MPDU). The protocol allows using both of them simultaneously but for the sake of simplicity, the driver used in the work presented uses only A-MPDU because of its more robust performance against poor channel conditions as compared to that of A-MSDU. Both of these mechanisms, originally conceived and implemented separately, are deeply intertwined and affect each other's behavior.

2.1.1 Aggregated MAC Protocol Data Units (A-MPDU)

802.11n facilitates aggregation of multiple MPDUs at the egress of MAC layer into an aggregated MPDU. The idea is to aggregate multiple frames, so as to avoid the overhead of preamble and header information when the channel is good by essentially treating multiple MAC layer MPDUs as one single MPDU. There is an upper limit of 10 ms on the amount of time that one single PPDU can use the medium for [9]. Hence, the number of MPDUs in a given aggregate (i.e. aggregation level) is a function of the physical layer transmission rate; the lower the rate, the fewer aggregated frames in an A-MPDU. There is also an upper limit on the aggregation level imposed by the receiving station's capabilities and communicated at the time of setting up the block acknowledgment session. In the experiments we performed, the limit is 64 frames, which is the maximum as per the 802.11n amendment.

2.1.2 Block Acknowledgment

Block acknowledgment is the mechanism which augments frame aggregation. The term refers to a wide variety of evolving techniques that are relevant to the 802.11 realm, but in the context of this thesis, we use the immediate block acknowledgments protocol as illustrated in Figure 2.1 for sessions of transmissions. To share parameters like traffic identifier, type of block acknowledgment (BA) mechanism, and starting sequence number (SSN) of the frames sequence that will be included in the session, the transmitting station establishes a block acknowledgment session by sending an add block acknowledgment (ADDBA) request to the receiving station, which is acknowledged. Upon receiving the ADDBA response from the receiver and sending the acknowledgment, the station starts transmission of A-MPDU and receives a bitmap as a part of block acknowledgment.

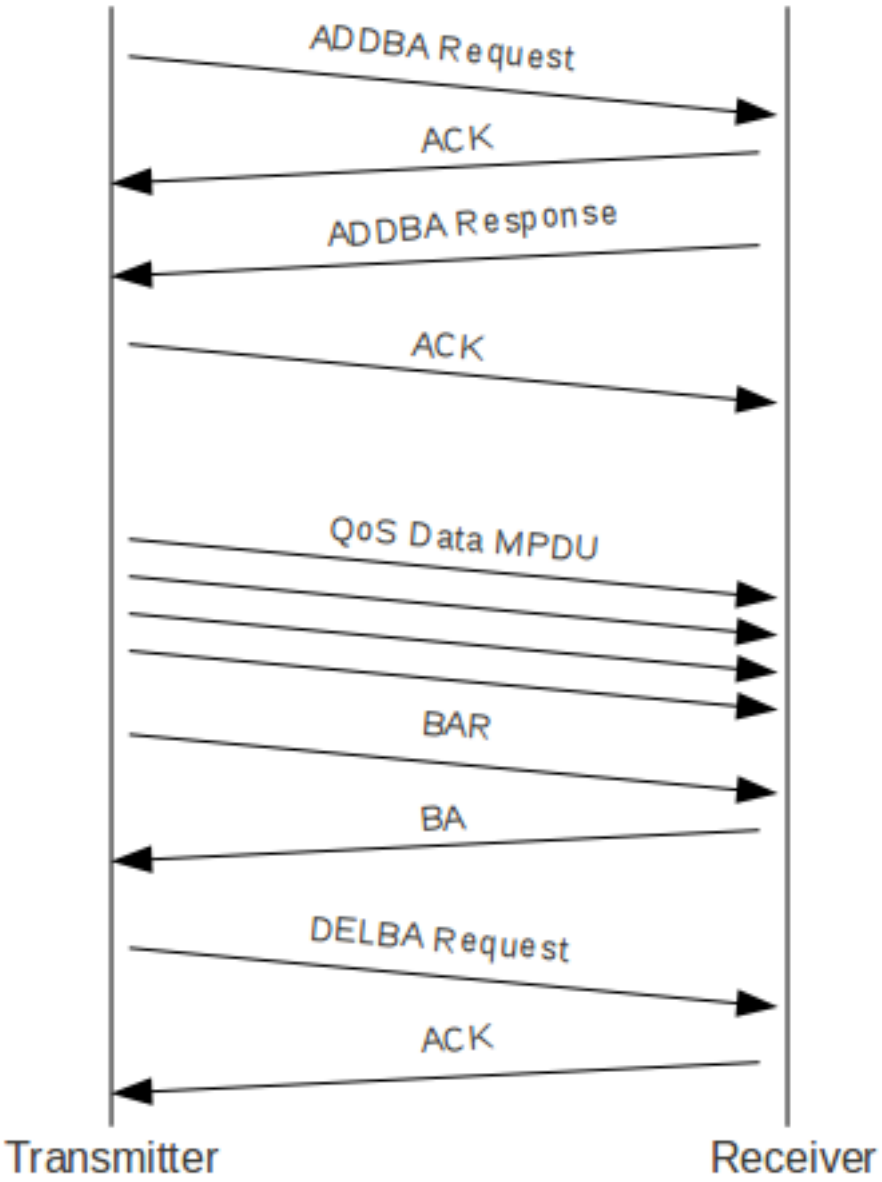

Figure 2.1: Block acknowledgment mechanism in 802.11n

This bitmap received in the BA frame contains information about the success or failure of previously transmitted frames, in order. If a certain frame experiences failure, it is re-queued for re-transmission with the next batch. There is a moving window being mantained at a transmitter which is called the block acknowledgment window. It can be thought of as the current context of frames being considered for transmission. These are the frames for which the transmitter will expect the acknowledgment. Frames with

different sequence numbers are numbered with a given index within a block acknowledgment window; however, at a given time, the range of sequence numbers that the block acknowledgment window can correspond to is fixed to a maximum of 64 in our experiments. The session is closed with a delete block acknowledgment (DELBA) message sent by the transmitter, which is acknowledged by the receiver.

2.2 Hardware and Ath9k Driver

Four machines are used in our experiments: two netbooks and two desktops. All four use wireless cards with Atheros chipset on them. The wireless cards used on desktops have Atheros chipsets AR5008 supporting dual spatial stream rates, and the internal cards on laptops have Atheros chipsets AR9285 and they only support single spatial stream rates. All of these cards operate in the 2.4 GHz band. The selection of these cards was based upon availability of reasonably modifiable driver code.

Ath9k is one of the mac80211 API based open-source wireless drivers for systems running Linux kernels. It supports a variety of chipsets including the ones used in the experiments performed in this study. As evident from the architecture in Figure 2.2 the driver is decomposed in several smaller pieces. Most of the MAC layer management entity (MLME) is implemented in the mac80211 module; however, the ath9k module is the one that performs aggregation. We modified the driver to allow for exporting stats using debugfs [10].

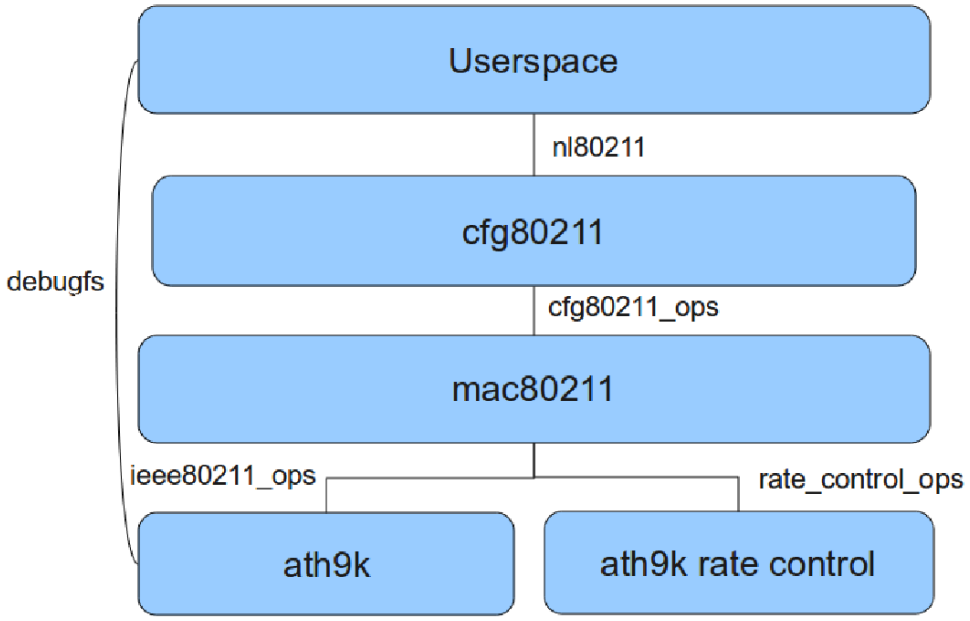

Figure 2.2: Ath9k architecture

2.3 Supported Physical Layer Rates

The hardware that we are using supports only a subset of the physical rates specified by 802.11n in various modes [9]. There are two broad modes in which the cards can be configured: 20 MHz channel mode, called HT20, and 40 MHz channel mode, called HT40. Both of these modes are mandated to support single MIMO stream rates, which is what all the hardware cards we used in our experiments do, whereas the standard provides up to quad-stream rates. The desktop PCI cards we used in our experiments support up to dual stream rates.

As presented in Tables 2.1 and 2.2, the rates categorized according to the number of spatial streams they use in the MIMO PHY. The standard also specifies rates based on a short inter-OFDM-symbol guard interval (GI), but the hardware used in our experiments only offers one dual-stream rate with that capability.

Table 2.1: HT20 Rate Table Employed During the Experiments

Modulation	**Coding Rate**	**DataRate (Mb/s)**	**Number of Streams**
BPSK	1/2	6.5	1
QPSK	1/2	13	1
QPSK	3/4	19.5	1
16-QAM	1/2	26	1
16-QAM	3/4	39	1
64-QAM	2/3	52	1
64-QAM	3/4	58.5	1
64-QAM	5/6	65	1
BPSK	1/2	13	2
QPSK	1/2	26	2
QPSK	3/4	39	2
16-QAM	1/2	52	2
16-QAM	3/4	78	2
64-QAM	2/3	104	2
64-QAM	3/4	117	2
64-QAM	5/6	130	2

Table 2.2: HT40 Rate Table Employed During the Experiments

Modulation	**Coding Rate**	**DataRate (Mb/s)**	**Number of Streams**
BPSK	1/2	13.5	1
QPSK	1/2	27	1
QPSK	3/4	40.5	1
16-QAM	1/2	54	1
16-QAM	3/4	81	1
64-QAM	2/3	108	1
64-QAM	3/4	121.5	1
64-QAM	5/6	135	1
16-QAM	1/2	108	2
16-QAM	3/4	162	2
64-QAM	2/3	216	2
64-QAM	3/4	243	2
64-QAM	5/6	270	2
64-QAM	5/6	300	2 (400ns GI)

2.4 Anechoic Chamber

Part of the experiments performed in this thesis were conducted in the Illinois Wireless Wind Tunnel's anechoic chamber [11]. An anechoic chamber provides a controlled environment for experiments by shielding the space inside it from all electromagnetic radiations outside the chamber, which means it is an interference-free environment. Inside, the chamber is lined with radiation absorbing foam panels, which reflect minimal energy; hence the walls inside the chamber act as energy absorbers, thus producing no multipath. So, essentially, the losses inside the anechoic chamber are the result of low SNR. Figure 2.3 is a photograph from an experiment instance; the two netbooks used in the experiments can be seen to have been placed at the maximum possible distance from each other in the chamber.

It would be interesting to see the behavior of dual stream MIMO rates within this chamber, which theoretically should not provide any gain because

of lack of multipath; but due to unavailability of requisite mobile hardware, we deferred that to future work.

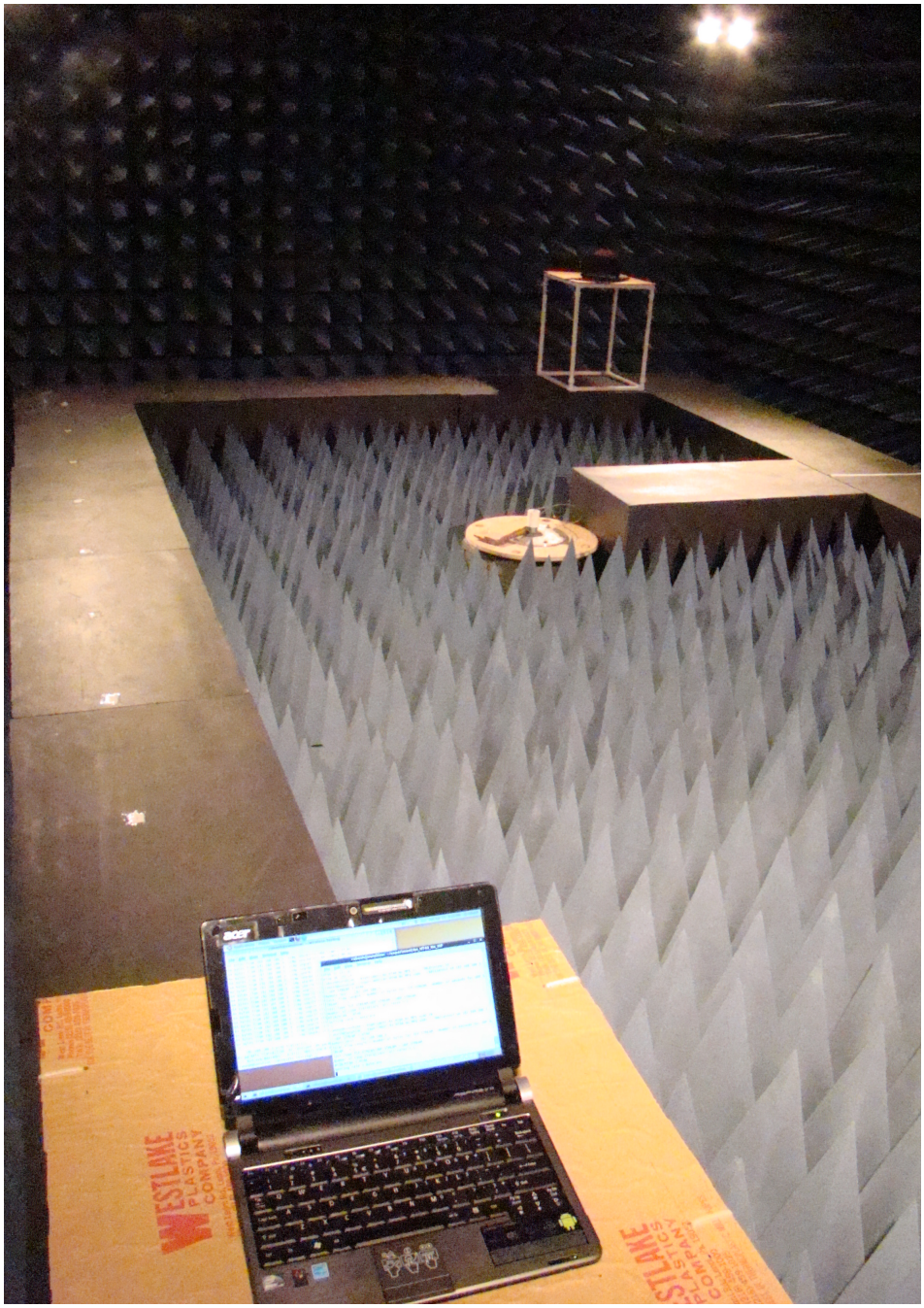

Figure 2.3: Experiment conducted in the anechoic chamber

CHAPTER 3

RELATED WORK

This chapter describes the ath9k's default rate control algorithm and provides some results from previously conducted measurement studies for 802.11bgn networks.

3.1 Ath9k Rate Control Algorithm

Minstrel [5] is used as the default algorithm for ath5k and madwifi drivers. It makes no assumptions about patterns and probability of frame losses and considers them to be an entirely random function. Mistrel design is based on sending probe frames to assess the condition of channel and likelihood of success of a given rate at any given time. The probing rate is variable but set to a default of 10 probes per second, so the slot-length of measurement is 100 ms. It computes and maintains the following expected-throughput metric for every rate at the end of a slot:

$$T_{r,t} = \frac{P_{r,t} \times B_r}{\tau_r} could \tag{3.1}$$

$$P_{r,t} = \alpha \times P_{r,t-1} + (1 - \alpha) \times P_{r,t-2} \tag{3.2}$$

$P_{r,t}$ is observed, successful frame transmission rate and $T_{r,t}$ is expected throughput for rate r in time-slot t. B_r is the number of bytes that have been transmitted using rate r, and τ_r is the amount of time it takes to transmit a single frame using rate r. Parameter α is used to control how much past results matter.

Minstrel exploits the multi rate retry (MRR) feature provided on Atheros and other chipset vendors. MRR is essentially an optimization feature provided by chip vendors to enhance the rate-adaption agility of their products. There is a retry-chain, whereby one can specify up to five rates and the num-

ber of times each rate is to be attempted by the hardware, in case of failure. Thus, minstrel would fill in the retry chain as depicted in Table 3.1.

Table 3.1: Rate Filling Strategy Used by Minstrel During MRR

Try	Probing		Not Probing
	Random $< \arg\max_r T_{r,t}$	Random $\geq \arg\max_r T_{r,t}$	
1	$\arg\max_r T_{r,t}$	Random r	$\arg\max_r T_{r,t}$
2	Random r	$\arg\max_r T_{r,t}$	$\arg\max_r T_{r-1,t}$
3	$\arg\max_r P_{r,t}$	$\arg\max_r P_{r,t}$	$\arg\max_r P_{r,t}$
4	Lowest Rate	Lowest Rate	Lowest Rate

The default algorithm for ath9k [12] is essentially based on ideas that are used in Minstrel. There are few 802.11n specific modifications whereby the fraction of unacknowledged frames over the total frames in the BA in the previous A-MPDU transmission are considered as a parameter to update the $P_{r,t}$. The parameter α is fixed to $\frac{1}{8}$ and, unlike Minsrtel probing, is fixed to happen every 50 ms.

3.2 Measurement Studies

Previously done measurement studies on 802.11n [2] and [13] indicate that frame aggregation and block acknowledgment mechanisms are crucial to achieve the performance gains for which the standard was ratified.

COLLIE [14] attempts to classify the cause of frame losses incurred in previous transmissions. It accomplishes that by asking the receiver to relay the last frame received in error along with RSSI back to the transmitter. It then uses metrics like RSSI, BER, symbol error rate (SER), error per symbol (EPS) and symbol error score (SES) to classify the cause of loss as either a loss due to weak signal or a loss due to interference. The authors of [14] propose to adapt the rate only in response to frame loss caused by weak signal and use the 802.11 backoff mechanism to fight interference. This simplistic, static threshold-based classification is shown to have improved throughput by 20-60%.

In RBAR [1], Wong et al. discover that the mutual information in the failure events of equally sized frames separated by up to 150 frames in 802.11b networks is higher than zero. Also they find that the consecutive frame transmission failure and success event have a sharply decreasing CDF beyond two success/failure events. This leads us to ask how the error patterns in an aggregate are correlated in the presence of an interference pattern generated by an 802.11g/n source.

In the measurement study employing 802.11 a/b/g conducted by Reis et al. [8] and in one employing 802.11n conducted by Shrivastava et al. [13], for a given transmitter, the conditional probability that the ith frame is lost given that $(i-k)$th frame is non-zero decreases with increasing k, until it reaches the unconditional frame loss probability at around $k = 150$. According to the findings in Reis et al. [8], the conditional probability also has a cycle of spikes with period corresponding to beacon frames.

CHAPTER 4

EXPERIMENTS

We decided to set up black-box experiments to explore the performance of ath9k rate control. Along with that, we measured how different BA parameters, as described in the ensuing sections, vary across different physical layer rates in different channel conditions. Experiments were performed in both controlled and uncontrolled environments on commodity hardware. The process that was used to prepare the machines for experiments and results is given in the following sections. Following that, we describe the parameters that we chose to look at from the block acknowledgment bitmap statistics and present the resulting measurement results.

4.1 Enabling Process

The first step of any measurement study is to prepare the hardware and software that shall be used. Our requirement was to be able to perform the experiments in an office lab as well as an anechoic chamber. For hardware we settled on commodity PCI 802.11n wireless cards installed in desktop machines with Pentium (D) processors at 2.66 GHz and 1 GB RAM, and two netbook machines with 1.66 GHz Atom processors and 1 GB RAM with 802.11n wireless cards built in. The operating system of choice was Ubuntu 8.10 with kernel 2.6 on all machines. The choice of hardware was motivated by availability. However, it required that the supporting device driver can be modified according to the required changes. The only compatible and modifiable device drivers available to us was ath9k. The following sections elaborate on the precise modifications that the device driver had to undergo to be useful in our study and describe the batch process script.

4.1.1 Driver Modifications

The open-source ath9k driver does not come with the feature set that is essential to a measurement study such as ours. However, it does provide support for debugfs, which allows for some of simple modifications to happen. Debugfs is a RAM-based file system which allows the driver to read and write data to userspace files. We have used this feature to establish a configuration option for fixing the physical layer data rate of the interface and to export statistics to the userspace.

Fixing Physical Layer Rate

In the rate control component of the driver, multiple, two-dimensional arrays of rate tables are maintained. Each one of these arrays corresponds to a particular operating mode of the hardware, i.e. n, g, HT20, HT40, etc. The probing-based rate control algorithm has a hook with the mac80211 code in the function ath9k_rc_get_rate. At the end of execution of this hook prior to the start of a given transmit sequence hook, the tx_info struct is populated for the given skb. To fix the physical layer rate, we found a point in the code where we could safely populate the tx_info struct from userspace code. The code modification checks if the debugfs is indicating usage of a fixed rate and does so, if required. This modification is depicted graphically in Figure 4.1

Disabling Multi-Rate Retry

As has been previously mentioned in Section 3.1, the ath9k default algorithm makes use of the multi-rate retry (MRR) feature. But for getting a fair assessment of the performance of protocol, we chose to disable the MRR. The ath9k driver provides access to the memory location on the chipset whereby the number of retries to be attempted can be set. This memory location can be accessed via DMA, and by manipulating that the rate control algorithm picks the set of rates for the next transmission.

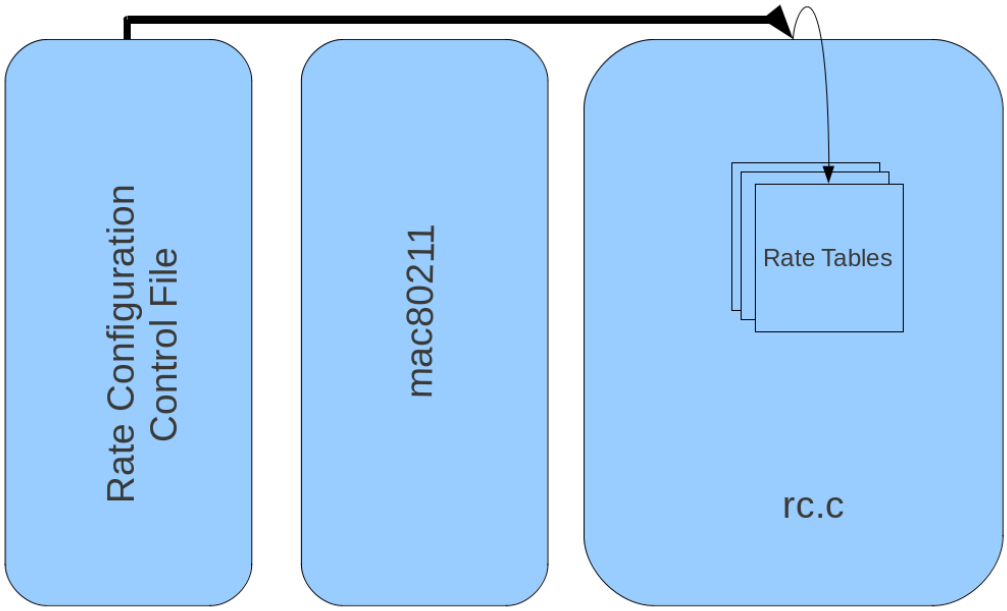

Figure 4.1: Driver architecture diagram for fixing physical layer rate

Exporting Block Acknowledgment Bitmap Statistics

Block Acknowledgment bitmaps, as explained previously, are returned as a part of the ACK frame from the receiver. In transmitter code of the driver, the transmit-ready frames are queued in the transmit buffers through DMA with the Atheros chip. Once the block acknowledgments are received for previously scheduled frames, the information is provided to the driver via an interrupt tasklet. We copy that information into a temporary RAM file via debugfs and constantly read this file in the userspace. As soon as the contents of the file are read, the file is emptied for more statistics. This modification is shown in Figure 4.2.

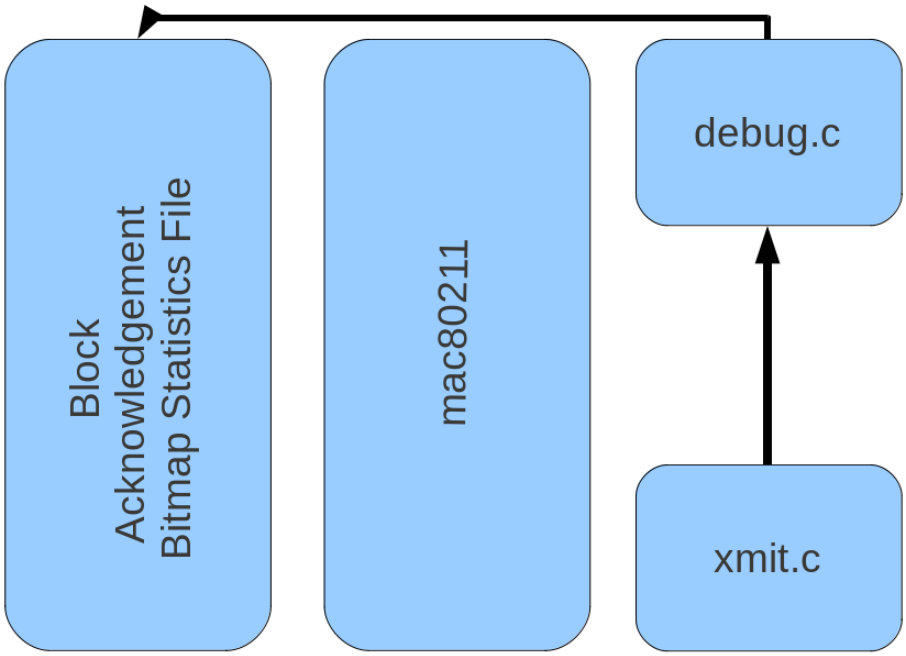

Figure 4.2: Driver architecture diagram for exporting block acknowledgment bitmap statistics

It must be noted that the code governing the generation of the block acknowledgment bitmap at the receiver is hidden and is hard-coded on the Atheros chips. This limits the possibility of using this same hardware to modify the BA mechanism.

4.2 Script Description

Both of the modifications described in previous sections are used by the batch process script. The userspace script has to essentially perform two overarching functions. One is to configure the interface physical layer data rate and the second is to collect BA bitmap statistics and analyze them. The script takes as input the operation mode (HT20/HT40), the number of times the experiment is to be performed and the length of UDP/TCP flow in seconds. All of these inputs are then used to select the relevant configuration parameters. In a cascaded loop among overall number of iterations and over physical layer data rates – including autorate related to the given operating mode – flows of TCP/UDP are executed in series one-by-one on all rates,

with the specified length. After the execution of the flow, the statistics are collected from the debugfs file and given to a bash script, which then sorts out the relevant parameters about the properties of the previous execution and summarizes them for plotting.

Netperf was used in the script to generate one second UDP flows, and the driver was configured to use the rate-control algorithm of ath9k and fixed PHY rate, one by one, in 40 MHz band, and resulting throughput results were averaged over 100 runs. The machines were configured to connect on a single link in infrastructure mode. The first set of experiments were performed on two static, desktop hosts put in an office environment both at day and night. The second set contained a single link between two netbooks having 802.11n enabled wireless cards. However, these wireless cards did not support any dual-stream (MIMO) rates. These experiments were performed in two settings: first in the office environment, and then in the anechoic chamber [11] to observe performance in an interference-free environment.

4.3 BA Bitmap Parameters

Upon initial analysis of the resulting BA bitmap statistics, we noticed that there were clusters of failed and successful transmissions. The cluster size, along with the size of the total bitmap, would be variable, the latter especially being a function of the transmission rate. So we decided to devise some parameters that could summarize the nature of the statistic and give a feel for what the results are. After some deliberation, we settled on the parameters explained below.

Normalized, Average Number of Transitions

In a given BA bitmap, this is the number of times a successful transmission was preceded by an failed one. For example, for a BA bitmap such as 1001001111, where 1 and 0 represent successful and failed frame transmissions, this number would be 2. This number indicates roughly the amount of burstiness of failures. We normalize the value in every BA bitmap and take an average over a given run. This metric roughly indicates the amount of channel variation in the previous transmission.

Normalized, Average Zero and One Burst Sizes

In a given BA bitmap, the average numbers of consecutive successful (1) or failed (0) transmissions are referred to as average one and zero burst sizes respectively. For example, 11001111000000001 has average zero burst size of 5 and average one burst size of 3.5. We normalize the value in every BA bitmap and take an average over a given run. This metric roughly indicates how good or bad the channel has been in the previous BA, and combined with the metric above, it indicates the channel burstiness.

Fraction of Good Frames

In a given BA bitmap, the total numbers of good and bad frames are also collected. We use a metric that is the fraction of good frames over the total number of frames in the previous BA. The metric is used directly by the ath9k default rate control algorithm and is used in the performance analysis for the same.

4.4 Measurement Results

In the subsequent subsections, we summarize what our observations were and the interpretations.

4.4.1 Ath9k Rate Control Algorithm Does Not Converge to Optimal Rate

The primary function of a rate control algorithm has been established to converge to the optimal rate under varying channel conditions. To gauge the performance of the default rate control algorithm of ath9k, we plot the CDF of the UDP throughput measured across all 100 iterations using the autorate algorithm as well as a few high performing rates in Figures 4.3, 4.4 and 4.5. The legends of these figures indicate the physical layer data rates in Mbps.

In the case of the desktop link in office environment and netbook link in both office environment and anechoic chamber, there is always at least one rate which consistently yields higher throughput than that is achieved using

the autorate algorithm. This clearly indicates that the ath9k default rate control algorithm consistently fails to converge to the optimal rate.

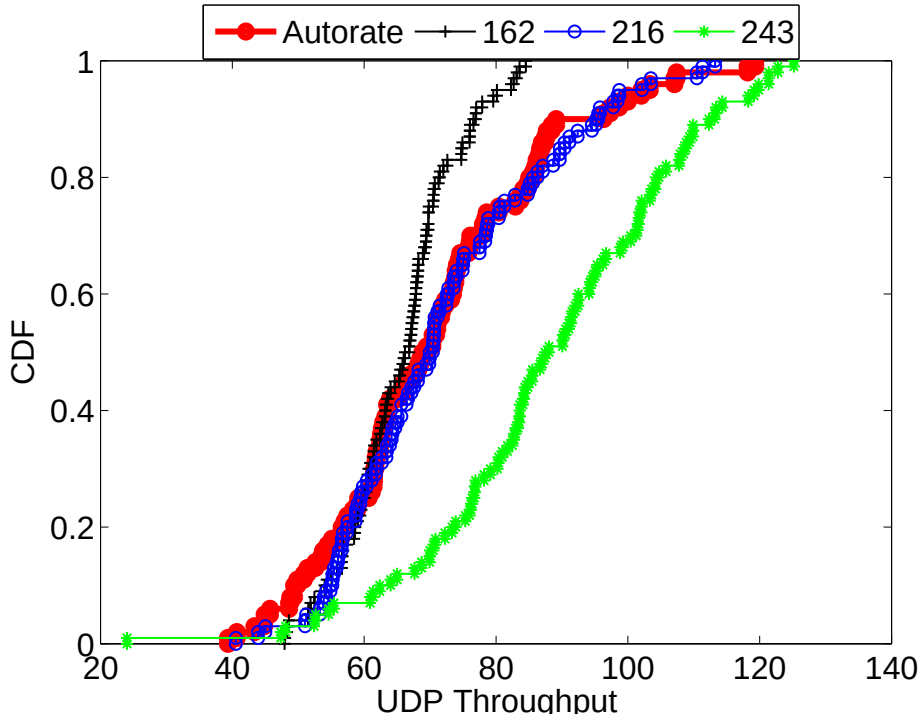

Figure 4.3: Distribution of observed UDP throughput (Mbps) on the desktop link in office environment

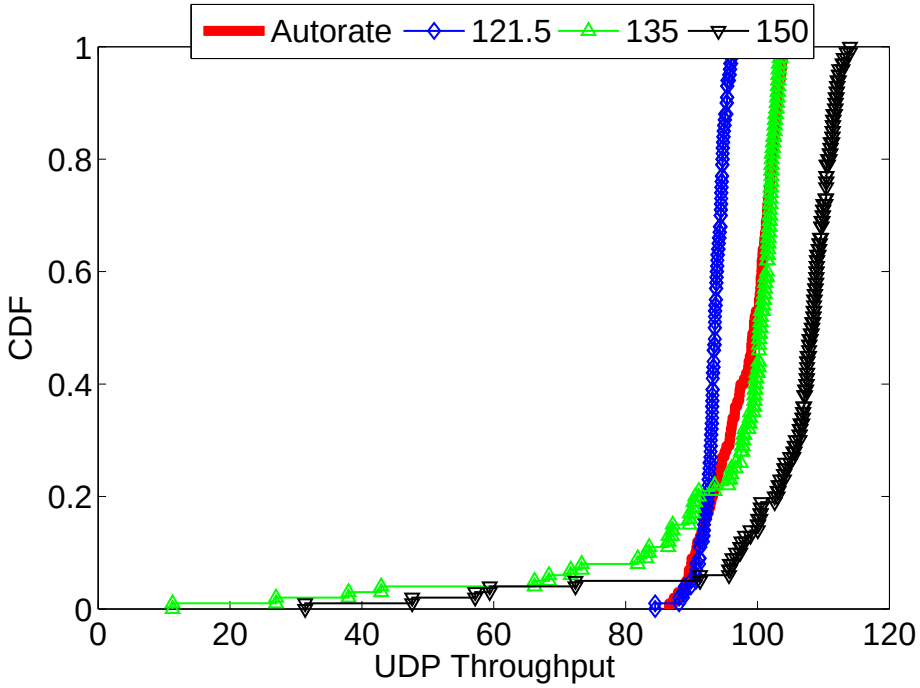

Figure 4.4: Distribution of observed UDP throughput (Mbps) on the Netbook link in office environment

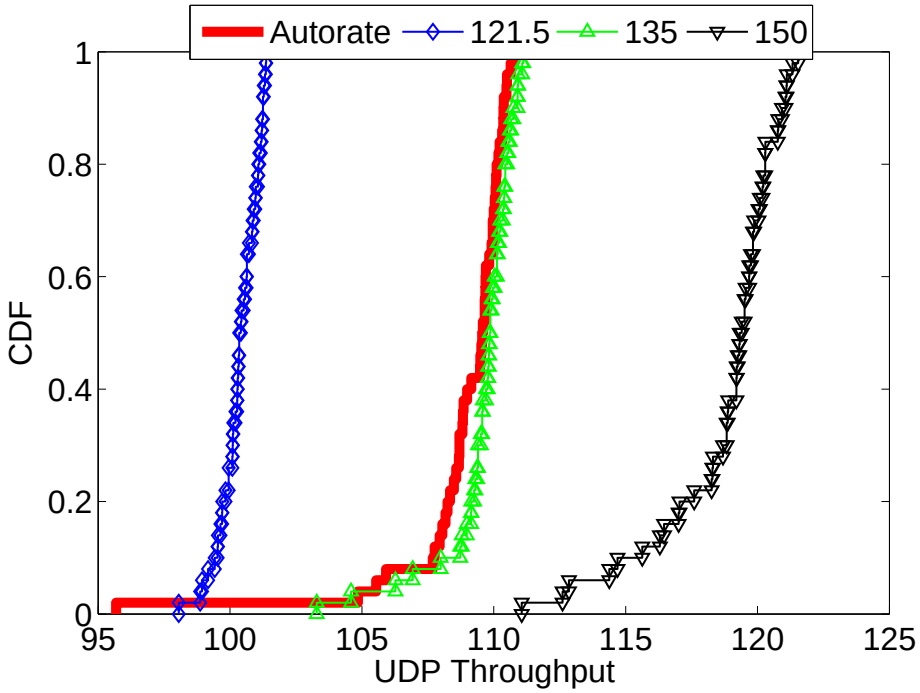

Figure 4.5: Distribution of observed UDP throughput (Mbps) on the netbook link in anechoic chamber

As previously stated in Section 3.1, the ath9k rate control algorithm attempts to take into account the fraction of failed frames in BAs to converge to the best rate for a given channel condition. For further investigation of the exact cause of such behavior, we plot the distribution of the average fraction of bad frames in BAs for the desktop link in office in Figure 4.6. The results indicate that while the ath9k rate control algorithm successfully optimizes the fraction of bad frames, it fails to provide higher throughput because different rates have different numbers of total frames, in BA so simply optimizing the fraction does not guarantee the absolute amount of data that is successfully transmitted on the link. Wong et al. [1] offer explanation of pitfalls of probing based rate control algorithms in general.

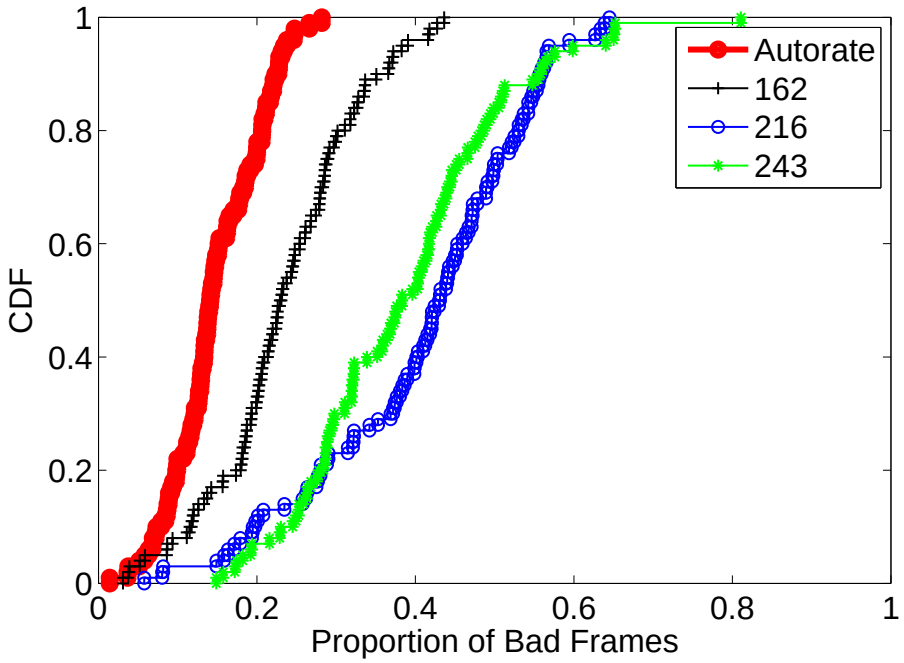

Figure 4.6: Distribution of fraction of bad frames in BA over desktop link in office environment

4.4.2 BA Bitmap Patterns Follow Channel Variations

As previously noted, the anechoic chamber provides a multi-path-free and interference-free channel. Our goal was to establish whether the BA bitmap values really represent the variations in the channel. So, Figures 4.7 and 4.8, we provide a comparison of average, normalized number of transitions and

average one burst sizes in both office and anechoic chamber environments, labeled OF and AC respectively in the legends.

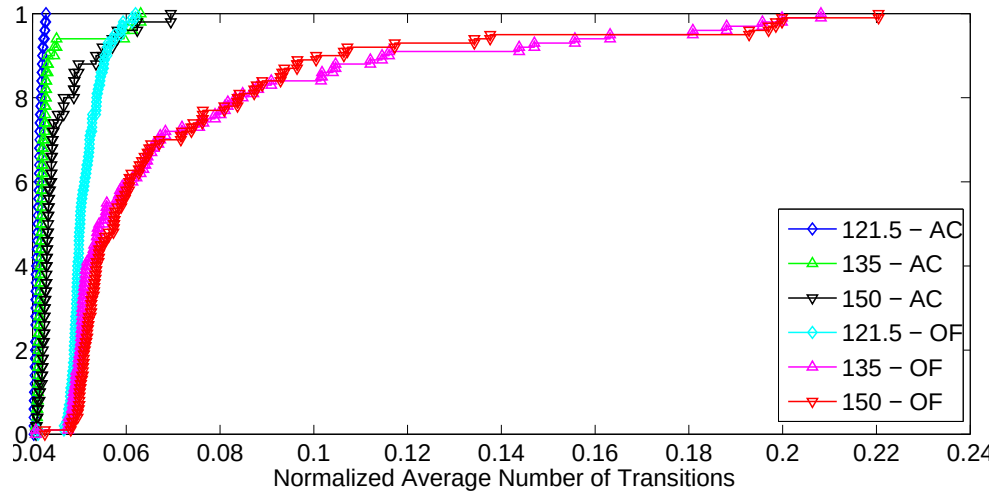

Figure 4.7: Comparison of average, normalized number of transitions in office and anechoic chamber

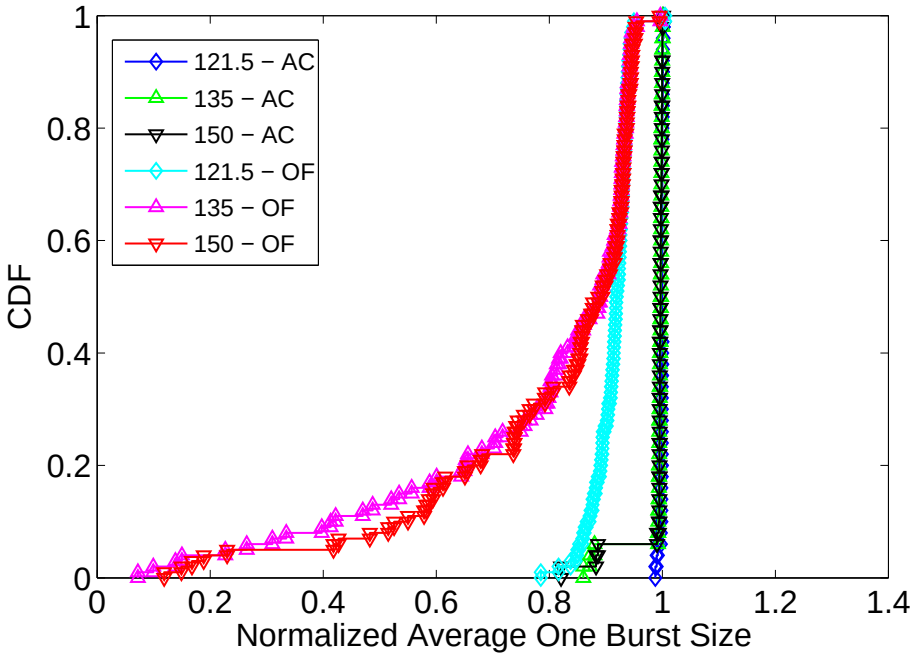

Figure 4.8: Comparison of average, normalized one burst size in office and anechoic chamber

The distributions in the figures indicate that office environment suffers

higher number of transitions in general and there is a long tail in their distribution which may be explained by interference. Meanwhile the metrics inside the anechoic chamber stay constant, following a more coherent channel.

4.4.3 Peak Performing Rates Are Different in Day and Night

In the HT40 mode on the desktop link in the office environment, we looked for the rates that give the highest throughput during day and night. In both cases, we find that best performing rates are dual stream rates, but the exact order of their performance changes. In Figure 4.9, we characterize this by plotting the CDF of UDP throughputs on three highest throughput yielding rates during the day and night.

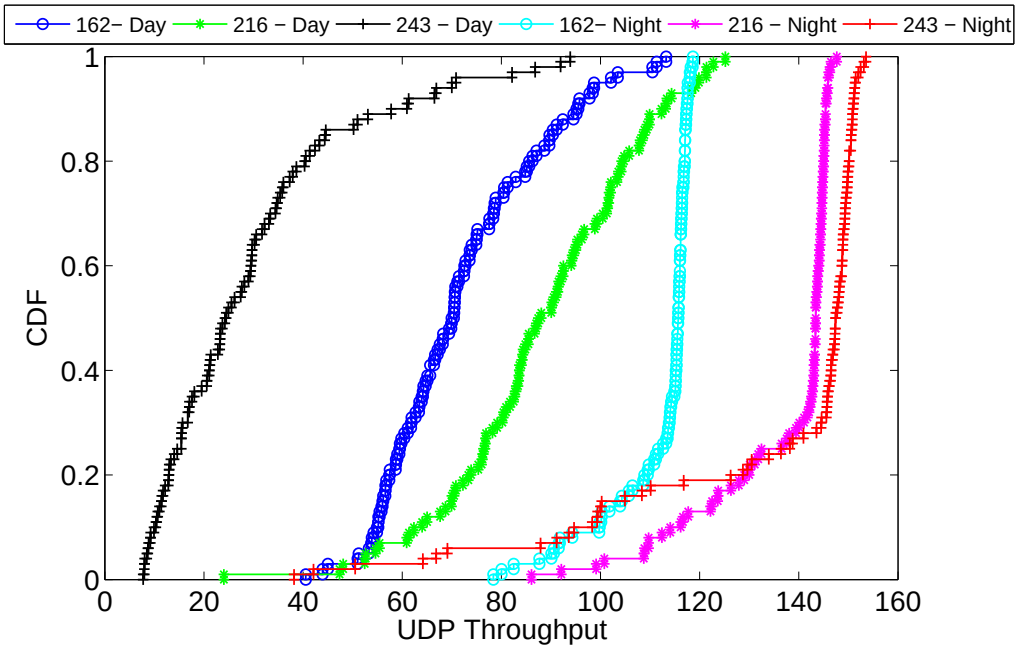

Figure 4.9: Comparison of Distribution of highest throughput (Mbps) rates during day and night

This indicates that the peak potential throughput performance of the link is higher at night than at day time. Our hypothesis as to the reason for this is that the channel conditions are better at night than during the day, i.e., there is lack of external interference. We further characterize these findings by presenting the distribution of collected BA bitmap statistics in Figures 4.10, 4.11 and 4.12.

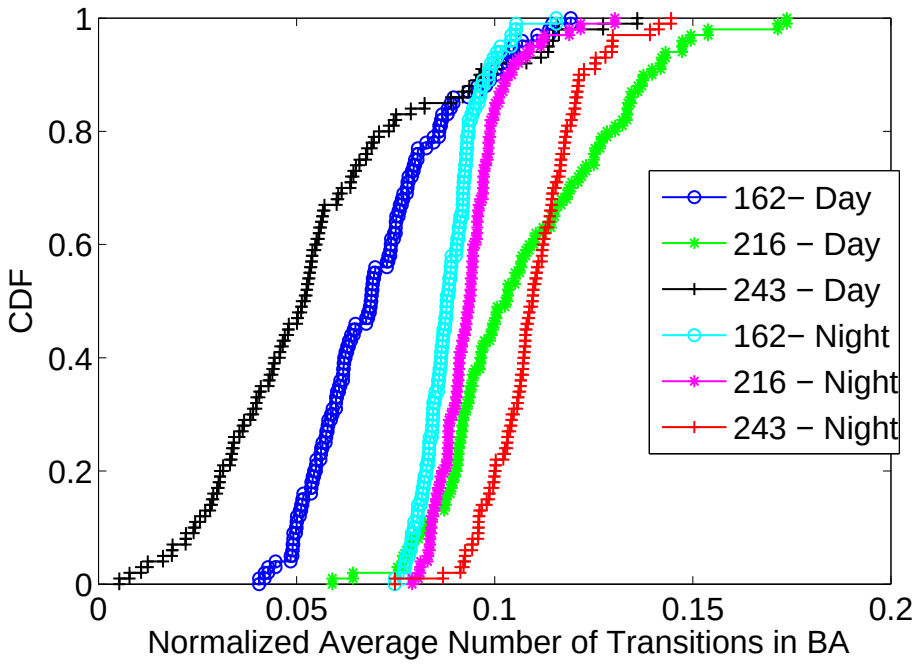

Figure 4.10: Comparison of distribution of normalized, average number of transitions during day and night

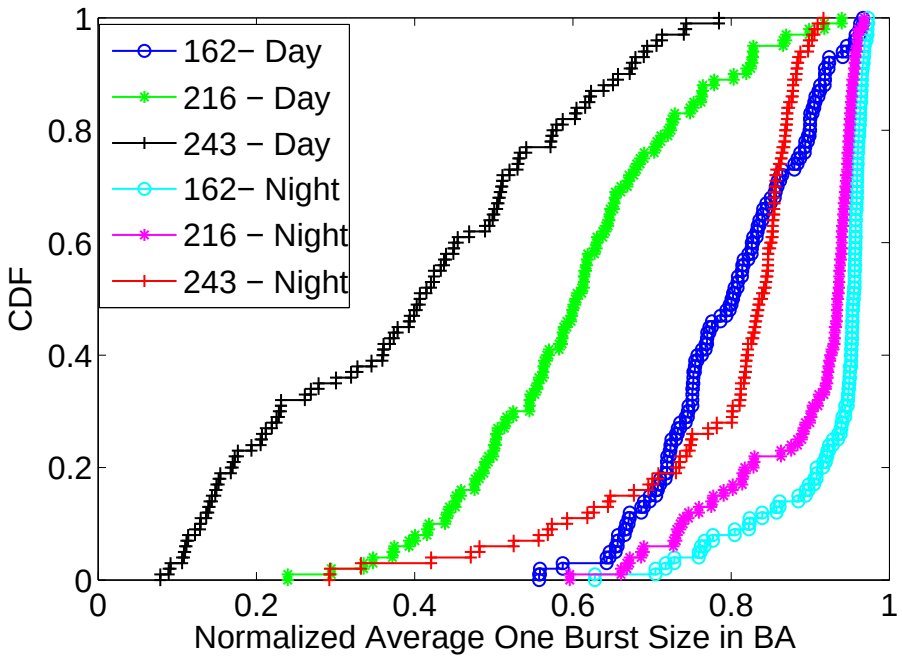

Figure 4.11: Comparison of distribution of normalized, average one burst size during day and night

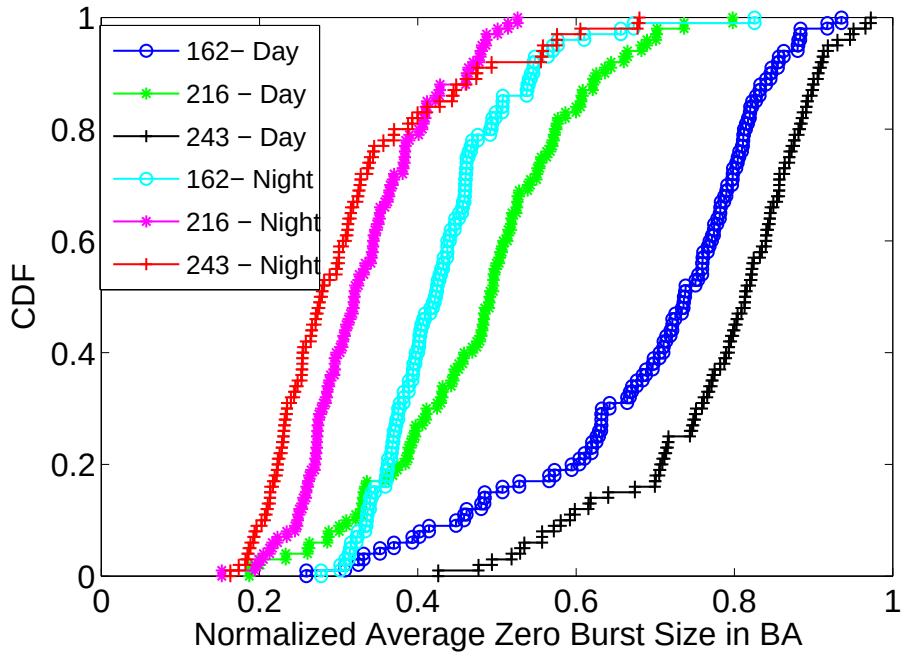

Figure 4.12: Comparison of distribution of normalized, average zero burst size during day and night

4.4.4 Channel Variations Heavily Influence the Performance of Spatial Streams

In the HT40 mode, there are two rates which both correspond to physical layer data rate of 108 Mbps, but they differ in the number of spatial streams, coding rate and modulation order, the details of which are given in Table 2.2. Figure 4.13 plots the distribution of UDP throughput achieved by each one of these rates in the office environment over the desktop link during both day and night. We chose the time difference because the channel is prone to less interference and channel variations in the office environment at night.

As evident from Figure 4.13, the single stream rate achieves marginally higher throughput than the dual stream rate, during the day. However at night, dual stream rate achieves marginally higher throughput. Also, as Figures 4.14 and 4.15 indicate, single stream rate has higher average number of transitions in BA and lower average one burst size at night. Hence one could argue that in the absence of external interference at night, single stream rate performs worse than the dual stream counterpart in the throughput sense.

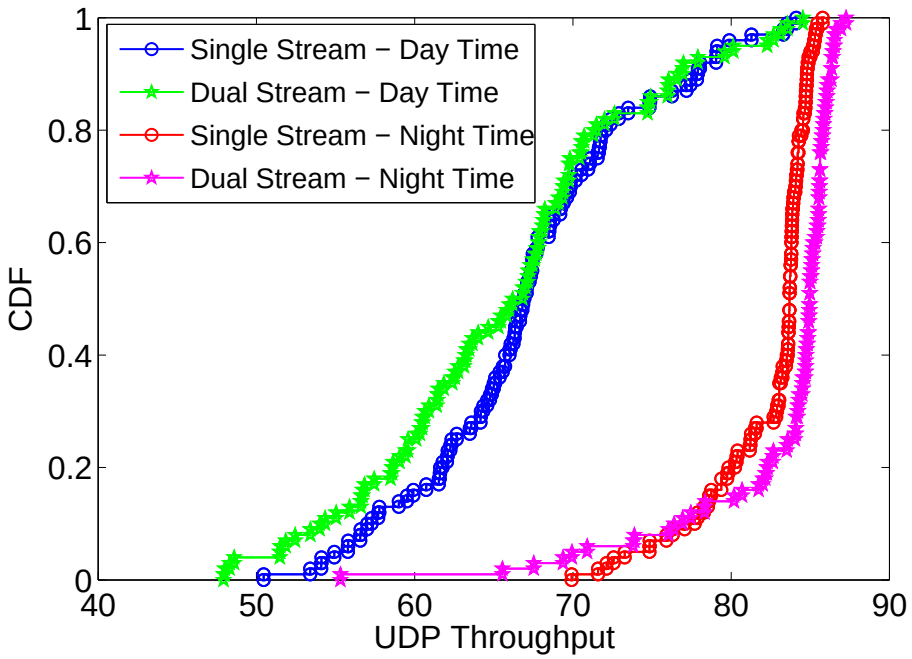

Figure 4.13: Comparison of distribution of observed UDP throughput (Mbps) between a single stream and dual stream rate

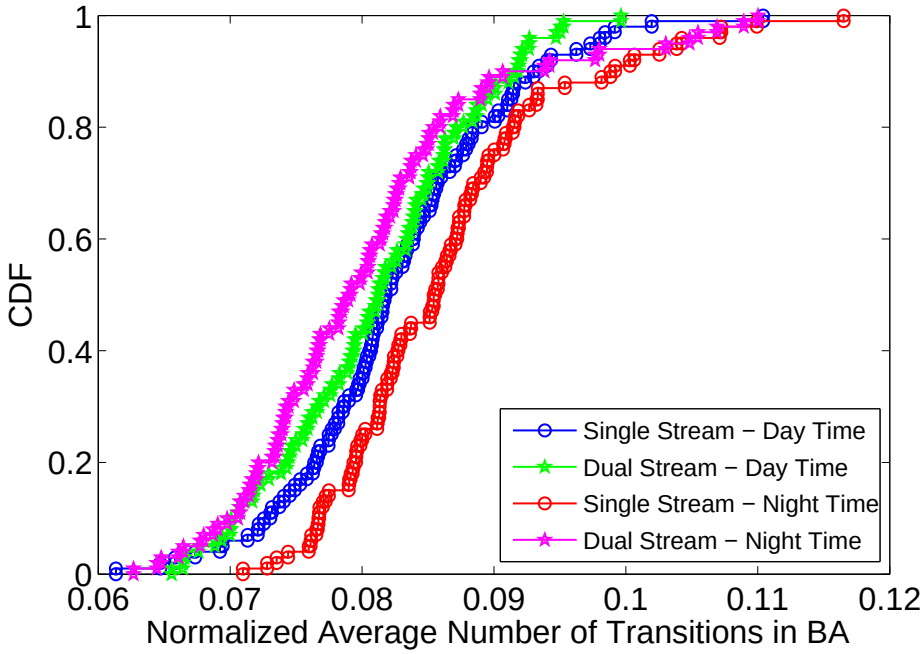

Figure 4.14: Comparison of distribution of normalized, average number of transitions between a single stream and dual stream rate

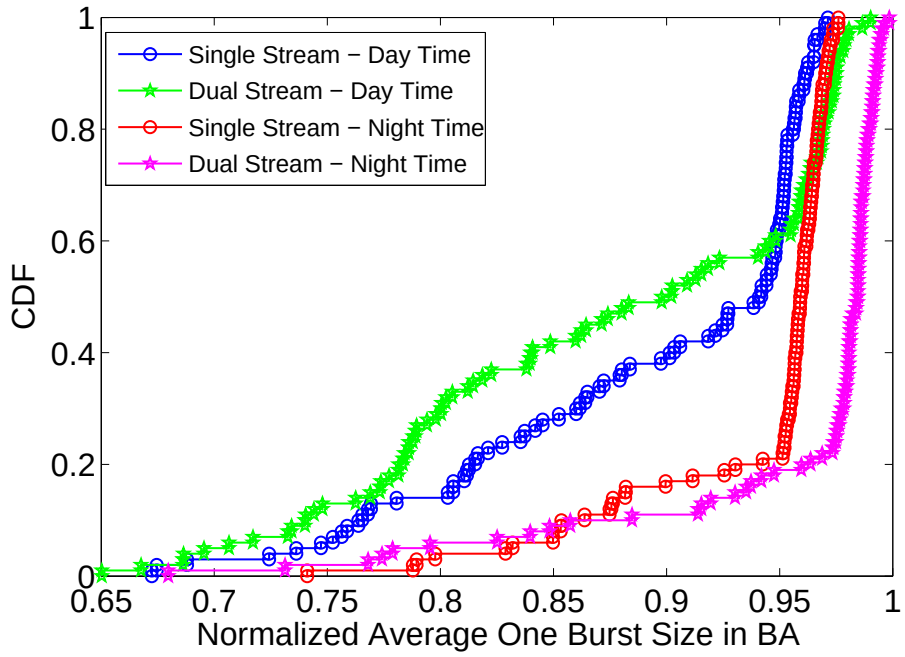

Figure 4.15: Comparison of distribution of normalized, average one burst size between a single stream and dual stream rate

In this chapter, we described the process of developing the experimental setup. We explained the key metrics that we chose to look at from the data collected in several channel conditions and compared the results.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize our findings and speculate on a possible rate control algorithm design for 802.11n networks on commodity hardware.

5.1 Summary of Measurements

In this study, we find that the ath9k driver's default rate control algorithm does not converge to the optimal rate under given channel conditions in virtually all possible scenarios. We devise several metrics and show by experiments that they follow channel variations, as we anticipated. We use these devised metrics to further study the variation in link throughput in several scenarios. We compare them in the office environment and discover that the peak performing rates are different during day and night. We further show how physical layer rates incorporating single or dual spatial streams perform in the office environment. We also provide comparison of link performance in anechoic chamber and compare that with the same link in the office environment.

5.2 Future Work

The overarching goal of this study has been to find design guidelines to develop a better rate control algorithm for 802.11n networks on commodity hardware. It can be seen in the measurement results in Chapter 4 that the normalized average burst size is correlated to throughput, unlike the fraction of errors. A possible rate control algorithm design can explore the metrics that we developed in this study and optimize one or more of the metrics while attempting to converge to the best rate under given channel conditions. This is speculation, however. A rigorous design would have to consider larger

network sizes than a single link, but we expect the ideas found in this study to apply to larger networks, too.

With larger network sizes, however, one must precisely characterize the interference present in the medium. Our study goes as far as using anechoic chamber to nullify external interference, but a possible rate control algorithm should quantify the amount and nature of interference and carefully incorporate its presence in the design. One strategy to solve this problem at design time is to explore simulation and then perform experiments to verify the findings of simulation.

Also, most of prevalent analytic models [7], [15] consider the failure events at nodes to be an i.i.d. process, and for large enough network sizes, the predicted results are fairly close to those observed experimentally and through simulations. However, when considered individually, nodes' transmission success is probabilistically related to the previous transmissions. Especially, with the same order of time to transmit, with 802.11n aggregated frames, this becomes crucial because, in a given transmission, any knowledge or heuristic about the relationship of error patterns in successive transmissions can prove beneficial in devising a rate control strategy.

Most importantly, if one views a rate control algorithm as a chain of actions, the series of choices of actions in 802.11n is not straightforward as in 802.11bg networks. If the previous transmission experienced interference, then 802.11n can choose to go to a lower rate having more diversity (i.e., more antennas); however, if it is was not interference then one could go for a rate which does not necessarily use spatial diversity but uses the additional MIMO streams for multiplexing. But this again requires a very careful understanding of interference in an 802.11n network.

REFERENCES

[1] S. H. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *MobiCom '06: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. ACM, 2006, pp. 146–157.

[2] K. Pelechrinis, T. Salonidis, H. Lundgren, and N. Vaidya, "Analyzing 802.11n performance gains," *ACM SIGCOMM 2009 Conference on Internet Measurement*, 2009.

[3] E. Perahia and R. Stacey, *Next Generation Wireless LANs*. Cambridge, UK: Cambridge Univ. Press, 2008.

[4] M. Vutukuru, H. Balakrishnan, and K. Jamieson, "Cross-layer wireless bit rate adaptation," in *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, 2009, pp. 3–14.

[5] D. Smithies, "Linux wireless - minstrel," 2005. [Online]. Available: http://wireless.kernel.org

[6] M. Vutukuru, H. Balakrishnan, and K. Jamieson, "Cross-Layer Wireless Bit Rate Adaptation," in *ACM SIGCOMM*, Barcelona, Spain, August 2009.

[7] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 535–547, March 2000.

[8] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-based models of delivery and interference in static wireless networks," in *SIGCOMM '06: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. New York, NY, USA: ACM, 2006, pp. 51–62.

[9] *IEEE 802.11n Draft Standard. Amendment 5*, IEEE Std. 802.11n, 2009.

[10] J. Corbet, "An updated guide to debugfs," 2009. [Online]. Available: http://lwn.net/Articles/334546

[11] N. H. Vaidya, J. Bernhard, V. V. Veeravalli, P. R. Kumar, and R. K. Iyer, "Illinois wireless wind tunnel: A testbed for experimental evaluation of wireless networks," in *E-WIND '05: Proceedings of the 2005 ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis*. ACM, 2005, pp. 64–69.

[12] F. Fietkau, "Linux wireless - ath9k," 2010. [Online]. Available: http://wireless.kernel.org/en/users/Drivers/ath9k

[13] V. Shrivastava, S. Rayanchu, J. Yoonj, and S. Banerjee, "802.11n under the microscope," in *IMC '08: Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2008, pp. 105–110.

[14] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee, "Diagnosing wireless packet losses in 802.11: Separating collision from weak signal," in *INFOCOM 2008. The 27th Conference on Computer Communications*, April 2008, pp. 735–743.

[15] Y. Lin and V. Wong, "Frame aggregation and optimal frame size adaptation for IEEE 802.11n WLANs," in *Global Telecommunications Conference, 2006. GLOBECOM '06*, Nov. 2006, pp. 1–6.